The code used to recreate the fivethirtyeight's approval graph is divided into 4 parts:

1. Loading libraries
2. Loading data
3. Data wrangling
4. Plotting

Part 1 - Loading Libraries

For this assignment, we use the followring libraries: tidyverse – an umbrella library that includes many important R libraries such as dplyr and ggplot – and ggthemes. From tidyverse, dplyr will be especially useful to us in the wrangling part, and ggplot is utilized for plotting. Here is the code used to load the libraries:

```
# loading libraries
library(tidyverse);
library(ggthemes);
```

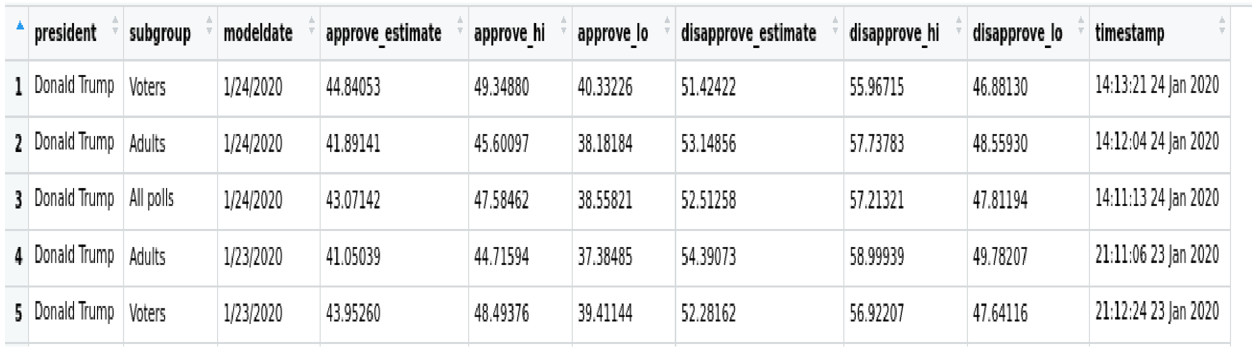
Part 2 - Loading Data

This is the a straightforward "read" step, where we use the function read.csv() to read in the data from the online source. Here is the code for that part:

```
# loading data
approval_topline <- read.csv("https://projects.fivethirtyeight.com/trump-approval-
data/approval_topline.csv");
```

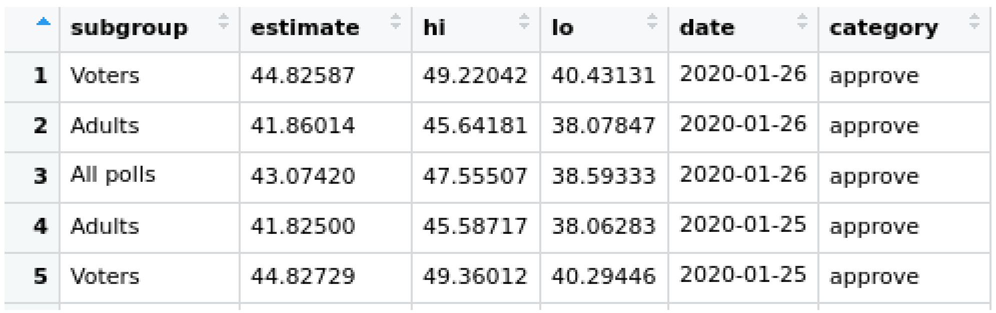
Part 3 - Data Wrangling

Next, we modify the shape of the original data, to something that we actually plot. The new data will take the shape of a table that has the same observations as the original, but instead of having all the approve/disapprove information (by that I mean the estimate, hi, and lo columns) shown in a single row, the new table will have them split to "approve" & "disapprove", with the respective highs and lows for each. This will allow us to plot the approval estimates in the same plot. The end result will be a dataframe that has 6594 entries instead of the 3297 in the original data (the reason being the split we just discussed). The new dataframe won't have neither the "president" nor the "timestamp" columns (the reason will be explained below). Finally, it will have a new attribute called "category", populated with distinct values for the "approve" & "disapprove" observations. The picture below shows both the original and new dataframes:



	president	subgroup	modeldate	approve_estimate	approve_hi	approve_lo	disapprove_estimate	disapprove_hi	disapprove_lo	timestamp
1	Donald Trump	Voters	1/24/2020	44.84053	49.34880	40.33226	51.42422	55.96715	46.88130	14:13:21 24 Jan 2020
2	Donald Trump	Adults	1/24/2020	41.89141	45.60097	38.18184	53.14856	57.73783	48.55930	14:12:04 24 Jan 2020
3	Donald Trump	All polls	1/24/2020	43.07142	47.58462	38.55821	52.51258	57.21321	47.81194	14:11:13 24 Jan 2020
4	Donald Trump	Adults	1/23/2020	41.05039	44.71594	37.38485	54.39073	58.99939	49.78207	21:11:06 23 Jan 2020
5	Donald Trump	Voters	1/23/2020	43.95260	48.49376	39.41144	52.28162	56.92207	47.64116	21:12:24 23 Jan 2020

The original dataframe



	subgroup	estimate	hi	lo	date	category
1	Voters	44.82587	49.22042	40.43131	2020-01-26	approve
2	Adults	41.86014	45.64181	38.07847	2020-01-26	approve
3	All polls	43.07420	47.55507	38.59333	2020-01-26	approve
4	Adults	41.82500	45.58717	38.06283	2020-01-25	approve
5	Voters	44.82729	49.36012	40.29446	2020-01-25	approve

The new dataframe. Notice the category column at the end

Now we go through the steps we need to go through in order to create this new dataframe:

1. Drop the “president” & “timestamp” columns, the former for being useless – since it has only one value – the latter for being redundant – I will be relying on the column “modeldate” for date referencing. This is done using the subset() function, which as the name indicates, provides a subset of the data it takes in based on some condition. It takes in the name of the dataframe it will be subsetting (df), along with the condition, which is defined by the select parameter. We use the -c() to not select the columns we’ll be dropping.

2. Create a new column called “date” which has the same values as the ones in “modeldate” but in date type instead of string, and drop the “modeldate” column. Using the as.Date command to change the string value to date type, and then dply’s select with the contain option to drop the “modeldate” column*.

3. Create a new dataframe that has all the columns minus the ones that have the word “disapprove” in their column name

4. Repeat step 3, but this time the discarded columns are the ones that have the word “approve”

5. Rename the columns in the 2 newly created dataframes, so that both dataframes will have matching column names (with will be needed for step xxx)

6. Add a new column called “category” to both dataframes, and fill it with the values “approve” & “disapprove”

7. Finally, bind the 2 dataframes by row, i.e. on top of one another, to have a new dataframe that has all the observations sorted by approve/disapprove**.

Here is the code used to do the above:

```
# data wrangling
df <- subset(approval_topline, select = -c(president, timestamp));
df$date <- as.Date(df$modeldate, "%m/%d/%Y");
df <- subset(df, select = -c(modeldate));
df_a <- select(df, -contains("dis"));
df_d <- select(df, -approve_estimate, -approve_hi, -approve_lo);
df_a <- rename(df_a, estimate = approve_estimate, hi = approve_hi, lo = approve_lo);
df_d <- rename(df_d, estimate = disapprove_estimate, hi = disapprove_hi, lo = disapprove_lo);
df_a$category <- "approve";
df_d$category <- "disapprove";
df <- rbind(df_a, df_d);
```

* PS 1: I was unable to do this step using lubridate commands

** PS 2: I tried using gather to achieve a similar result, but the _hi & _lo columns were being repeated for both types of observations, so I resorted to this method

Part 4 – Plotting

The plot consists of 7 parts:

A) The base

B) The line

C) The smooth

D) The ribbon

E) The pointrange

F) The label

G) The annotation

H) The theme, which in our case was fivethirtyeight

We’ll go through each one of them in details:

A- The Base:

It is the foundation of our plot. It contains the data we’re about to plot, and it is here that we set the axis values. We will be creating 4 different plot, a combined plot showing the approval/disapproval rates for all subcategories in a vertical grid, and a separate plot for each subcategory. An important thing to note here is that the base, defined by the ggplot() function, is not an actual plot, rather, it needs to be combined with the other pieces in order to produce an actual figure. For our plot, the ggplot() takes in 2 parameters: the data and the aesthetics. The dataset will differ for each plot, the grid plot will take in the complete dataset, while the separate plots will take a filtered version of the dataset , something we achieve using the filter() function.

B- The Line:

The centerpiece of the plot, and the first producible plot. The line function doesn't take any parameters.

C- The Smooth:

This is a smoothed version of the line. As with the line function, no parameters are specified to the smooth() function.

D- The Ribbon:

The ribbon adds the hi and lo estimates to the main estimate, by specifying the ymin & ymax parameters in the aes(). We use the fill parameter to ensure that the added lines will have a similar color to the lines we plotted, to avoid visual jittering.

E- The Pointrange:

Pointranges are data points plotted over the line, with vertical lines connecting the point to the hi & lo lines. It is a visual aid that shows interval confidence at a given point. To avoid cluttering, we will subset the data by taking samples every 6 weeks (instead of daily data points). We do this using the filter() function combined with as.Date() - used to manipulate the date values. Inside the as.Date() function we'll set the breaks to "6 weeks". As with the base, we create 4 separate pointranges, as each will be working on different subset.

F- The Label:

This is the title given to the plot

G- The Annotation:

To annotate the plot, we will use the geom_text() function

I- The Theme:

This is a simple call to the theme_fivethirtyeight() function

Here is the code used to do the above:

```
# plotting
p <- ggplot(df, aes(x = date, y = estimate, color = rev(category)));
p_v <- ggplot(filter(df, subgroup == "Voters"), aes(x = date, y = estimate, color = rev(category)));
p_a <- ggplot(filter(df, subgroup == "Adults"), aes(x = date, y = estimate, color = rev(category)));
p_p <- ggplot(filter(df, subgroup == "All polls"), aes(x = date, y = estimate, color = rev(category)));
l <- geom_line() ;
s <- geom_smooth() ;
r <- geom_ribbon(aes(ymin = lo, ymax = hi, fill = rev(category), alpha = 0.001)) ;
pr <- geom_pointrange(data = filter(df, date == as.Date(cut(date, breaks = "6 week"))), aes(x = date,
ymin = lo, ymax = hi));
pr_v <- geom_pointrange(data = filter(filter(df, subgroup == "Voters"), date == as.Date(cut(date,
breaks = "6 week"))), aes(x = date, ymin = lo, ymax = hi));
```

```
pr_a <- geom_pointrange(data = filter(filter(df, subgroup == "Adults"), date == as.Date(cut(date,
breaks = "6 week"))), aes(x = date, ymin = lo, ymax = hi));
pr_p <- geom_pointrange(data = filter(filter(df, subgroup == "All polls"), date == as.Date(cut(date,
breaks = "6 week"))), aes(x = date, ymin = lo, ymax = hi));
lb <- labs(title = "Presidents approval and disapproval rates");
lb_v <- labs(title = "Presidents approval and disapproval rates from Voters");
lb_a <- labs(title = "Presidents approval and disapproval rates from Adults");
lb_p <- labs(title = "Presidents approval and disapproval rates from All polls");
g <- geom_text(data = filter(df, date == max(date)), aes(label = estimate), hjust = -0.1, vjust = +1)
g_v <- geom_text(data = filter(filter(df, subgroup == "Voters"), date == max(date)), aes(label =
estimate), hjust = -0.1, vjust = +1)
g_a <- geom_text(data = filter(filter(df, subgroup == "Adults"), date == max(date)), aes(label =
estimate), hjust = -0.1, vjust = +1)
g_p <- geom_text(data = filter(filter(df, subgroup == "All polls"), date == max(date)), aes(label =
estimate), hjust = -0.1, vjust = +1)
p + l + s + r + pr + lb + facet_grid(vars(subgroup)) + theme_fivethirtyeight() + theme(legend.position
= "none") + g;
p_v + l + s + r + pr_v + lb_v + theme_fivethirtyeight() + theme(legend.position = "none") + g_v;
p_a + l + s + r + pr_a + lb_a + theme_fivethirtyeight() + theme(legend.position = "none") + g_a;
p_p + l + s + r + pr_p + lb_p + theme_fivethirtyeight() + theme(legend.position = "none") + g_p
```